UNIVERSITY OF SASKATCHEWAN

Department of Computer Science

CMPT 215.3 MIDTERM EXAMINATION

November 6th, 2001

Total Marks: 50

CLOSED BOOK and CLOSED NOTES
NO CALCULATOR

Time: 75 minutes

Instructions

Read each question carefully and write your answer legibly on the examination paper. **No other paper will be accepted.** You may use the backs of pages for rough work but all final answers must be in the spaces provided. The marks for each question are as indicated. Allocate your time accordingly.

Ensure that your name AND student number are clearly written on the examination paper and that your name is on every page.

Note: a reference table of MIPS instructions is provided at the end of the examination paper.

Question	Marks
1 (5 marks)	4
2 (12 marks)	9
3 (18 marks)	10
4 (15 marks)	7
Total	30/50

Name: ______________________

Student Number: 930772

Name:______________________________

Student Number: 930772

1. **General** *(5 marks)* Give the technical term that best fits each of the following descriptions or definitions.

   (a) A special purpose register (on a MIPS system, not directly accessible to the machine language programmer) that stores the address of the next instruction to be executed.

   Accumulator ✗

   (b) An algorithm for multiplication of signed integers in which a run of consecutive "1"s in the multiplier is handled with just two arithmetic operations (plus some shifts): a subtraction of the multiplicand at the beginning of the run, and an addition of the multiplicand just after the end of the run. Booth's algorithm ✓

   (c) A program chosen to serve as the basis of performance comparison between computer systems. Benchmark ✓

4/5

   (d) A numbering system with 16 digits, which are represented by the symbols 0-9, and A,B,C,D,E, and F. Hexadecimal ✓

   (e) A law stating that $\overline{A} + \overline{B} = \overline{A \bullet B}$.

   De Morgan's law ✓

9

2. **Computer Performance** *(12 marks in total)*

   2 (a) (*2 marks*) Suppose that the MIPS rating for a particular program is ~~450~~ 300 and the clock rate is ~~300~~ 450 MHz. What is the CPI?

   $CPI = \frac{450}{300} = 1.5$ ✓

   $CPI = \frac{\text{clock rate}}{MIPS}$

Student Number: 930772

5 (b) (6 *marks*) In each of the following parts, state which ones of the three factors determining CPU execution time (number of machine language instructions executed, clock cycle time, CPI) may change, when the indicated system change is made (and all else is held constant).

2 (i) a clock with higher frequency is used

number of machine language instructions executed, clock cycle time.

3 (ii) a new machine language instruction is implemented for an operation that was previously done in software (i.e., with a sequence of simpler instructions); this is done without impacting the implementation of the existing instructions or their execution times

number of machine language instructions executed, CPI

0 (c) (*2 marks*) Give a formula for the *geometric mean* of three numbers T1, T2, and T3, and state one motivation for using the geometric mean (rather than the arithmetic mean) when computing a single number summarizing system performance.

arithmetic = $\frac{T1 + T2 + T3}{3}$

2 (d) (*2 marks*) Consider a system with two classes of instructions. Class *A* instructions have a CPI of 2, and class *B* instructions have a CPI of 8. Suppose that it is possible to reduce the CPI of class *B* instructions to 5, but at the cost of an increase in the clock cycle time. If 1/3 of the instructions in a particular program are of type *A*, and 2/3 are of type *B*, what would be the maximum factor by which the clock cycle time could increase, without increasing the program's execution time?

	CPI	new CPI
A	2	2
B	8	5

$4\overline{)6.0}$ 1.5

20

$CPU_{old} = (2 \times 1/3) + (8 \times 2/3)$

$= 2/3 + 16/3 = 18/3 = 6$

$CPU_{new} = (2 \times 1/3) + (5 \times 2/3) = 2/3 + 10/3 = 12/3 = 4$

The clock cycle time could increase about 1.5 times before the execution time would be different.

Name:______________________

Student Number: 930772

$2^5 = 32$

3. **Arithmetic** *(18 marks in total)*

   (a) *(8 marks)* Give the base 10 number that is represented by 111011, assuming each of the following representations:

      (i) 6 bit 2's complement
      $(-1\times2^5)+(1\times2^4)+(1\times2^3)+(1\times2^1)+(1\times2^0)$
      $= -32+16+8+2+1 = -5$

      (ii) 6 bit biased notation with bias of 31 $(x-31)$
      $32+16+8+2+1 = 59$
      $59-31 = 28$

      (iii) 6 bit 1's complement
      $000100 = 4$
      -4

      8

      (iv) 6 bit sign-magnitude
      $-(16+8+2+1) = -27$

   (b) *(2 marks)* How can one tell that a number in the IEEE 754 floating point standard format is a *denormalized* number? It is a denormalized number if there is any number but zero in front of the decimal point. 1.1×10^0 is normalized but 10.1×10^0 or 0.11×10^0 are not normalized.

      1

10/18

   (c) (*2 marks)* Consider the addition of 4 bit values $a_3a_2a_1a_0$ and $b_3b_2b_1b_0$. Give a logic function for the "carry-in" to the most significant bit position, in terms of quantities $g_i = a_i \cdot b_i$, $p_i = a_i + b_i$, and the "carry-in" to the least significant bit position $CarryIn_0$.

      $a_3a_2a_1a_0$
      $b_3b_2b_1b_0$

      Carry-in = $g_1 + g_2 \cdot g_3 \cdot g_4$

      Carry-in =

      X 0

Name:________________________________

Student Number: 930772

2
64
32
16
8
4
2
126

(d) *(2 marks)* Recall that in the IEEE 754 floating point standard, single precision floating point numbers have a 1 bit sign field, followed by an 8 bit exponent field (in biased notation with a bias of 127), followed by a 23 bit significant field. Give the number (in base 10) that is represented by $\boxed{1}\boxed{01111110}\boxed{1100000000000000000000.}$

$(-1)^S * F$

sign exponent-1
$-$ $(126-127)=-1$

$11/2^2 = 3/4$

1.75

-1.75×10^{-1}

1

(e) *(4 marks)* Give a **truth table** for a "3 data input, 1 output" multiplexor with *data* inputs a_0, a_1, and a_2, and *select* inputs s_0 and s_1. Suppose that a_0 always has the value 0 (so you don't need to show any rows in your truth table for a_0=1). Clearly state any assumptions you need to make. Then, using your truth table, derive a **logic equation** in sum-of-products form. (You **don't** need to simplify it.)

$s_0 = s_1 = 0$ then output =

a_0	a_1	a_2	s_0	s_1	output
0	1	1			1
0	0	1			1
0	1	0			1
0	0	0			0

X 0

output $= (\bar{a}_0 \cdot a_1 \cdot a_2) + (\bar{a}_0 \cdot \bar{a}_1 \cdot a_2) + (\bar{a}_0 \cdot a_1 \cdot \bar{a}_2)$

Student Number: 930772

7 4. **Machine and Assembly Language** *(15 marks in total)*

0 (a) *(4 marks)* What functions does a *linker* perform, and how does it carry out these functions?

A linker is an instruction that links two parts of a program so you can move to a new part of the program and then come back. It is also used in jump instructions and loops. It saves the address of the target and returns to the target once the operation is complete.

2 (b) *(2 marks)* Consider a proposed new instruction "memmov". For example, "memmov A, B", where A and B are symbolic names, would copy the contents of the memory location with address corresponding to B, to the memory location with address corresponding to A. What difficulty would arise if one tried to add this new instruction to MIPS machine language?

You would have to add a new field to MIPS that could handle two memory addresses.

Name: ______________________

Student Number: ~~Nikolay Raymond~~ 930772

110	111	112
2	2	2

4

(c) *(5 marks)* Consider the following code fragment.

```
        .data
        .align 2
A:      .word 2, 4, 6, 8, 10
B:      .word 1, 3, 5, 7, 9

        .text
main:   la $t0, A
        la $t1, B
        move $t2, $t1 # $t2 is the end address of array A
loop:   lw $t7, 0($t0)
        sw $t7, 0($t1)
        addi $t0, $t0, 4
        addi $t1, $t1, 4
        bne $t0, $t2, loop
```

2 (i) Following execution of the above code, what are the contents of the 10 words in the data segment? A: 2, 4, 6, 8, 10

B: 2, 4, 6, 8, 10

2 (ii) How would the result change, if at all, if the loop was changed to the following?

```
loop:   lb $t7, 0($t0)
        sb $t7, 0($t1)
        addi $t0, $t0, 1
        addi $t1, $t1, 1
        bne $t0, $t2, loop
```

The result would not change.

0 (iii) If the running time of the original loop is T, what is the running time of the loop in question (ii)? T/2. X

Name:

Student Number: 130772

1 (d) *(4 marks)* Procedure calls in MIPS should follow certain conventions. Answer the following two questions using the procedure call conventions discussed in class and in the text.

0 (i) Consider a procedure *foo* with five integer arguments. Supposing that a routine wants to call *foo* with the contents of \$s0, \$s1 and \$s2 for the first three parameters, and the values 3 and 7 for the 4th and 5th parameters, write a sequence of instructions that makes the call and passes the parameters according to the MIPS conventions.

1 (ii) Suppose that the procedure *foo* makes a call to another procedure, and then uses the result from that procedure call to compute its own return result. When computing its own return result, it uses registers \$s0 and \$t0 (changing the values in these registers). Write a sequence of instructions for use at the beginning of *foo*, that saves registers to the stack as is necessary.

\$sp

```
subi  $sp, $sp, 8
lw    $s0, 4($sp)
lw    $t0, 0($sp)
```

The End

MIPS machine language

Name	Format	Example						Comments
		6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	
add	R	0	2	3	1	0	32	add $1,$2,$3
sub	R	0	2	3	1	0	34	sub $1,$2,$3
addi	I	8	2	1		100		addi $1,$2,100
addu	R	0	2	3	1	0	33	addu $1,$2,$3
subu	R	0	2	3	1	0	35	subu $1,$2,$3
addiu	I	9	2	1		100		addiu $1,$2,100
mfc0	R	16	0	1	14	0	0	mfc0 $1,$epc
mult	R	0	2	3	0	0	24	mult $2,$3
multu	R	0	2	3	0	0	25	multu $2,$3
div	R	0	2	3	0	0	26	div $2,$3
divu	R	0	2	3	0	0	27	divu $2,$3
mfhi	R	0	0	0	1	0	16	mfhi $1
mflo	R	0	0	0	1	0	18	mflo $1
and	R	0	2	3	1	0	36	and $1,$2,$3
or	R	0	2	3	1	0	37	or $1,$2,$3
andi	I	12	2	1		100		andi $1,$2,100
ori	I	13	2	1		100		ori $1,$2,100
sll	R	0	0	2	1	10	0	sll $1,$2,10
srl	R	0	0	2	1	10	2	srl $1,$2,10
lw	I	35	2	1		100		lw $1,100($2)
sw	I	43	2	1		100		sw $1,100($2)
lui	I	15	0	1		100		lui $1,100
beq	I	4	1	2		25		beq $1,$2,100
bne	I	5	1	2		25		bne $1,$2,100
slt	R	0	2	3	1	0	42	slt $1,$2,$3
slti	I	10	2	1		100		slti $1,$2,100
sltu	R	0	2	3	1	0	43	sltu $1,$2,$3
sltiu	I	11	2	1		100		sltiu $1,$2,100
j	J	2			2500			j 10000
jr	R	0	31	0	0	0	8	jr $31
jal	J	3			2500			jal 10000

MIPS instruction formats

Name	Fields						Comments
Field size	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	All MIPS instructions 32 bits
R-format	op	rs	rt	rd	shamt	funct	Arithmetic instruction format
I-format	op	rs	rt	address/immediate			Transfer, branch, imm. format
J-format	op	target address					Jump instruction format

Main MIPS machine language. Formats and examples are shown, with values in each field: op and funct fields form the opcode (each 6 bits), rs field gives a source register (5 bits), rt is also normally a source register (5 bits), rd is the destination register (5 bits), and shamt supplies the shift amount (5 bits). The field values are all in decimal. Floating-point machine language instructions are shown in Figure 4.47 on page 291. Appendix A gives the full MIPS machine language.

MIPS operands

Name	Example	Comments
32 registers	`$s0-$s7, $t0-$t9, $gp, $fp, $zero, $sp, $ra, $at, Hi, Lo`	Fast locations for data. In MIPS, data must be in registers to perform arithmetic. MIPS register `$zero` always equals 0. Register `$at` is reserved for the assembler to handle large constants. `Hi` and `Lo` contain the results of multiply and divide.
2^{30} memory words	Memory[0], Memory[4], . . . , Memory[4294967292]	Accessed only by data transfer instructions. MIPS uses byte addresses, so sequential words differ by 4. Memory holds data structures, such as arrays, and spilled registers, such as those saved on procedure calls.

MIPS assembly language

Category	Instruction	Example		Meaning	Comments
Arithmetic	add	add	$s1,$s2,$s3	$s1 = $s2 + $s3	Three operands; overflow detected
	subtract	sub	$s1,$s2,$s3	$s1 = $s2 – $s3	Three operands; overflow detected
	add immediate	addi	$s1,$s2,100	$s1 = $s2 + 100	+ constant; overflow detected
	add unsigned	addu	$s1,$s2,$s3	$s1 = $s2 + $s3	Three operands; overflow undetected
	subtract unsigned	subu	$s1,$s2,$s3	$s1 = $s2 – $s3	Three operands; overflow undetected
	add immediate unsigned	addiu	$s1,$s2,100	$s1 = $s2 + $s3	+ constant; overflow undetected
	move from coprocessor register	mfc0	$s1,$epc	$s1 = $epc	Used to copy Exception PC plus other special registers
	multiply	mult	$s2,$s3	Hi, Lo = $s2 × $s3	64-bit signed product in Hi, Lo
	multiply unsigned	multu	$s2,$s3	Hi, Lo = $s2 × $s3	64-bit unsigned product in Hi, Lo
	divide	div	$s2,$s3	Lo = $s2 / $s3, Hi = $s2 mod $s3	Lo = quotient, Hi = remainder
	divide unsigned	divu	$s2,$s3	Lo = $s2 / $s3, Hi = $s2 mod $s3	Unsigned quotient and remainder
	move from Hi	mfhi	$s1	$s1 = Hi	Used to get copy of Hi
	move from Lo	mflo	$s1	$s1 = Lo	Used to get copy of Lo
Logical	and	and	$s1,$s2,$s3	$s1 = $s2 & $s3	Three reg. operands; logical AND
	or	or	$s1,$s2,$s3	$s1 = $s2 | $s3	Three reg. operands; logical OR
	and immediate	andi	$s1,$s2,100	$s1 = $s2 & 100	Logical AND reg, constant
	or immediate	ori	$s1,$s2,100	$s1 = $s2 | 100	Logical OR reg, constant
	shift left logical	sll	$s1,$s2,10	$s1 = $s2 << 10	Shift left by constant
	shift right logical	srl	$s1,$s2,10	$s1 = $s2 >> 10	Shift right by constant
Data transfer	load word	lw	$s1,100($s2)	$s1 = Memory[$s2+100]	Word from memory to register
	store word	sw	$s1,100($s2)	Memory[$s2 + 100] = $s1	Word from register to memory
	load byte unsigned	lbu	$s1,100($s2)	$s1 = Memory[$s2 + 100]	Byte from memory to register
	store byte	sb	$s1,100($s2)	Memory[$s2 + 100] = $s1	Byte from register to memory
	load upper immediate	lui	$s1,100	$s1 = 100 * 2^{16}	Loads constant in upper 16 bits
Conditional branch	branch on equal	beq	$s1,$s2,25	if ($s1 == $s2) go to PC + 4 + 100	Equal test; PC-relative branch
	branch on not equal	bne	$s1,$s2,25	if ($s1 != $s2) go to PC + 4 + 100	Not equal test; PC-relative
	set on less than	slt	$s1,$s2,$s3	if ($s2 < $s3) $s1 = 1; else $s1 = 0	Compare less than; two's complement
	set less than immediate	slti	$s1,$s2,100	if ($s2 < 100) $s1 = 1; else $s1=0	Compare < constant; two's complement
	set less than unsigned	sltu	$s1,$s2,$s3	if ($s2 < $s3) $s1 = 1; else $s1=0	Compare less than; natural numbers
	set less than immediate unsigned	sltiu	$s1,$s2,100	if ($s2 < 100) $s1 = 1; else $s1 = 0	Compare < constant; natural numbers
Unconditional jump	jump	j	2500	go to 10000	Jump to target address
	jump register	jr	$ra	go to $ra	For switch, procedure return
	jump and link	jal	2500	$ra = PC + 4; go to 10000	For procedure call

Main MIPS assembly language instruction set. The floating-point instructions are shown in Figure 4.47 on page 291. Appendix A gives the full MIPS assembly language instruction set.